Influences on Agile Practice Tailoring in Enterprise Software Development

Julian M. Bass
School of Computing
Robert Gordon University
Aberdeen, UK
j.bass@computer.org

***Abstract*—*Agile development projects have become a reality in large enterprises using offshore development models. A case study involving seven international companies with offices in Bangalore, India, and London, UK was conducted, including interviews with 19 practitioners. The contribution of this paper is to illustrate the reasons for tailoring Agile practices within the context of large enterprises. The findings show that scrum roles and practices did not conflict with enterprise policies or processes and were thought to improve product quality and productivity. However, agile practices from the XP tradition were not so widely adopted. Test driven development did not integrate well within enterprises where independent quality assurance teams were constituted as separate departments. Continuous integration was found to be challenging where enterprise software products required time consuming regression testing and elaborate code release processes. While adoption of coding standards and collective code ownership are necessary to facilitate interaction between disparate stakeholder groups.*

Keywords-enterprise software; distributed agile development; scrum; extreme programming (XP); tailoring

I. Introduction

Globalization in software development is driven by technological innovation, the evolution of work and business processes, as well as the prevailing education system and national policies [1]. This trend is irreversible and led by software intensive high-technology businesses [2]. In consequence, global software development has become the norm in large organizations. Simultaneously, enterprises need software applications that can: firstly, reconcile the (sometimes conflicting) needs of stakeholders, by secondly, implementing business policies and processes, that thirdly, bring value by fostering organizational goal attainment [3]. This must be achieved by enterprise software vendor companies that face a period of modest rates of growth and declining prices [4].

Agile practices can improve both software quality and team productivity. Tailoring of agile practices has been identified as necessary for their adoption in large organizations [5], or where traditional software development processes have previously been used [6]. The tailoring is required to interface to existing structures, processes and policies of the enterprise. This tailoring can include selection (or deselection) of specific agile practices or adaptation to integrate with aspects of the enterprise context over which individual project teams have little influence. Further, adaptation for geographically distributed implementation is required.

In this paper, a constellation of extreme programming (XP) and scrum practices using case studies are explored; including 19 practitioner interviews from 7 international companies. The investigation corroborates another study suggesting a surprising recent gain in momentum for Scrum practices [7]. However, the unexpected lack of support for XP practices in these companies was also notable. The following pages will describe reasons for this tailoring of practices in distributed agile development. But first, a more detailed discussion of previous research in this field is presented, starting with a brief overview of agile practices and more specifically those from Scrum and XP, followed by a discussion of distributed agile development.

A. Agile Software Development Processes

Agile software development practices improve responsiveness to customer needs, resulting in better software quality and improve team morale, resulting in enhanced productivity [8]. Practitioners report that the three most important perceived agile principles are: (1) achievement of customer satisfaction through early and continuous delivery of valuable software, (2) business representatives and developers working together frequently throughout the project, and (3) that face-to-face conversations are the most efficient way to convey information to, and within, the development team [9]. There is empirical evidence to support claims that agile methods improve job satisfaction, productivity and customer satisfaction but that the adoption by large and complex projects is difficult [10].

B. Scrum Roles and Practices

The focus of Scrum is on management techniques rather than on software development practices [11]. Scrum uses short iterations, lasting two to four weeks, called sprints. Software requirements are captured in the form of user stories. Scrum envisages a product owner who prioritizes user stories into a product backlog on behalf of the customer.

A sprint planning meeting is used to identify target functionality and estimate the time required for implementation within each sprint. Project team members communicate every day using a stand up meeting. The stand up meeting is intentionally short, 15 minutes, and each participant is required to address three questions: (1) what did you do yesterday? (2) what will you do today and (3) what blockers have you encountered or created for others? Blockers are issues that prevent progress.

Further, the scrum approach emphasizes software development in small increments using self-organizing, multidisciplinary "feature" teams [12] [13]. Feature teams concentrate on the holistic development of end-to-end functionality, or features, when seen from a user perspective. This stands in contrast, to approaches where functionality is hierarchically decomposed into architectural components, which are then developed by specialist teams (such as separate development teams for user interface and database software).

C. XP Practices

Extreme programming (XP) emphasizes skilful development practices and is founded upon the four values of: communication, simplicity, feedback and courage [14]. Building upon these values, twelve development practices are identified in XP: the planning game, small releases, metaphor, simple design, test driven development, re-factoring, pair programming, collective code ownership, continuous integration, 40-hour week, on-site customer and coding standards [15]. Notice that practices such as simple design, test driven development, re-factoring and continuous integration are focused very much on the software product itself. These practices illustrate how XP places more emphasis on software production techniques than Scrum.

D. Distributed Agile Development

The key issues in global software engineering is coordination over geographical distance [16]. Time separation, including: (1) time zones, (2) work day hours, (3) weekend working patterns and (4) public holiday differences are disruptive to global software engineering projects [17]. Minimization of this disruption is achieved through careful selection of synchronous and asynchronous communication methods. Collaborative software tools have been shown to increase access to awareness information for team members [18]. Yet, we still lack detailed understanding of the trade-offs between communication methods, software tools and software development practices [16].

Pilot projects exploring XP adoption in large organizations use tailoring of agile practices to fit particular organizational contexts [5]. Sometimes this tailoring can appear to outsiders as a traditional development process with a few agile practices used in specific circumstances. XP practices do not address problems encountered by multiple or distributed project teams [5]. Yet, there have been reports of success in the use of scrum in distributed agile projects [19]. Systematic review of the global software engineering research literature highlights continuous integration, standup meetings, pair programming, retrospectives, scrum of scrums and test driven development as the most popular agile practices reported [20]. However, it should be noted, that not all of this literature refers to large scale or complex enterprise projects.

More specifically, scrum teams use a wide range of team collaboration tools including: visits and periods of collocation working, unofficial meetings, training activities, distributed documentation support tools to help overcome sociocultural distance [21]. Scrum teams are supported with a wide range of tools to support multiple modes of communication including: phone, web camera, teleconference, video conference, web conference, net meeting, email, shared mailing lists, Instant Messaging (IM) and Short Message Service (SMS).

Having surveyed research in agile practices and distributed agile development relating to enterprise software development, the research methods used in this study can is now described. Information gathering techniques and participant selection is described in the following paragraphs. In Section III, the findings are presented highlighting the agile practices adopted and looking at the reasons for tailoring in an enterprise software context. The findings are discussed in relation to knowledge of the field from other research sources in Section IV. Limitations of the study are considered in Section V. Finally, in Section VI, conclusions and suggestions for further work are presented.

II. Research Methods

This research comprises a case study investigation into the adoption of distributed agile development practices in global enterprise software development projects [22]. The case studies involved investigation of publicly available as well as commercially confidential policy and project documents from the 7 international companies.

A combination of opportunistic and purposeful sampling was used to identify target projects. Opportunistic sampling was necessary since it is not feasible to "cold call" major global companies seeking to conduct case studies into their software development practices. A network of contacts from former work colleagues in the commercial sector and students of the Executive MBA program at the Indian Institute of Management-Bangalore were used to make contact with participating companies.

Face-to-face interviews were conducted with 19 practitioners, based in Bangalore, India and London, UK in January 2010 (11 interviews in Bangalore), February 2010 (3 interviews, in London) and April 2011 (5 interviews in Bangalore). The interviewees ranged from senior program managers, responsible for several large project teams through to experienced software developers. Interviewees included scrum masters, product owners as well as people describing themselves as architects and project managers, as shown in Table 1. A good level of data triangulation was achieved with a broad range of project stakeholders from Company E and their offshore software service provider Company D.

TABLE I. INTERVIEWEE OVERVIEW

	Sector	*Interviewee Job Titles*
Company A	IT Service Provider	Program Manager Senior Project Manager Team Member
Company B	Internet	Delivery Manager Product Manager (interviewed twice, Jan 2010 and April 2011)
Company C	Software Service Provider	Development Manager
Company D (Offshore Provider to Company E)	Software Service Provider	Project Manager Product Owner Scrum Master (3) QA Lead Team Member
Company E	Enterprise CRM	Program Manager Project Manager Director of Engineering
Company F	Industrial Products	Scrum Master
Company G	IT Service Provider	Engagement Manager

An open-ended interview guide approach was used [23]. There was some evolution of the interview guide during the three fieldwork study periods. A typical interview guide is summarized in Appendix 1. Interviews were recorded, transcribed and coded for analysis using an iterative process. Emergent themes were identified and incorporated into the coding scheme. Examples from the coding scheme include: project <identifier>, scrum role <scrum master, product owner, team member>, and agile practice <pair programming, test-driven development, shared ownership>.

III. FINDINGS

Discussion and advocacy of agile methods appear in the public documentation produced by the companies investigated here. It appears that agile project experience is an important capability for many offshore software service suppliers.

Analysis of interview transcripts shows that of the 19 practitioners interviewed, 2 indicated that they wanted to be regarded as having had bad experiences with agile methods. The remaining 17 interviewees could be regarded as agile advocates. Although interestingly, changes in job roles meant that 7 were not working on agile projects at the time of their interviews. The discussion of the findings presented here is in two parts: first, the frequency of adoption of agile practices, and second, the influences on adoption.

A. *Agile Practice Selection*

Some projects did not lend themselves to adoption of agile practices. The engagement manager at Company G contrasted software development projects and systems integration projects "*in product development we have a set of defined features. We know what we are going to do.*" He continues, "*In systems integration we need to interact with multiple vendors,... they don't have any knowledge of agile.*"

However, it is not surprising that evidence of agile practices was found, since there was some degree of purposeful sampling in the selection of companies to investigate. However, their choice of practices, in an enterprise software development context is interesting. Eleven of the projects surveyed here made extensive use of the roles and practices defined in the Scrum method, see Fig. 1. The underlying concepts of product owner, scrum master and self-organizing team were adopted by 11 projects. Similarly there was consensus on the key concepts of sprints, sprint planning and retrospectives. Notice on Fig. 1, that fewer projects provided evidence of the use of increment demos.

Short iterations are seen as a way of responding to market pressure, by reducing product release lead times. According to the scrum master at Company F "*in a traditional waterfall model, you go for a requirement analysis, then design, and then coding. And by the time you actually go for delivery ... the market situation has changed.*" Reducing time to market and getting feedback on product releases is important in the Internet domain of Company B "*[we want to] come in the market as soon as possible ... with newer ideas*" and "*you don't know your customers face-to-face, so it's pretty critical to get the product out [and] get their feedback.*"

The self-organizing team approach empowers team members, who have freedom to choose work tasks within a sprint and also make their own scheduling estimates. According to the product manager at Company B "*if people are being rotated based on their choice and interest their motivation level is higher ... and as a team our skills are going up.*" Reducing dependency upon specific individuals is valuable for reducing risk.

However, the implementation of the scrum practices was not entirely as envisaged by proponents (for example [13]). For example, several projects used Sprints but did not attempt to deliver functionality to clients each increment. In Company D, one month sprints produced code which was then collected over six to nine months and packaged into larger releases. Instead of producing shippable code, the emphasis of the sprint was to focus and motivate team members as a tool for productivity enhancement.

In contrast, few of the surveyed projects made use of XP practices, see Fig. 2. The software service provider, Company C, is an advocate of agile methods and makes extensive use of XP. Theirs was the only project in the study that aspired to fully implement XP style practices. According to the Director of Engineering at Company E:

> *We don't do extreme, definitely don't do extreme programming. ... we use scrum with Agile more as an organizational thing ... we meet in line with Agile process ... So all the administrative tasks are very much in line with what I understand to be Agile ... We don't use any things to improve the productivity of the team through different*

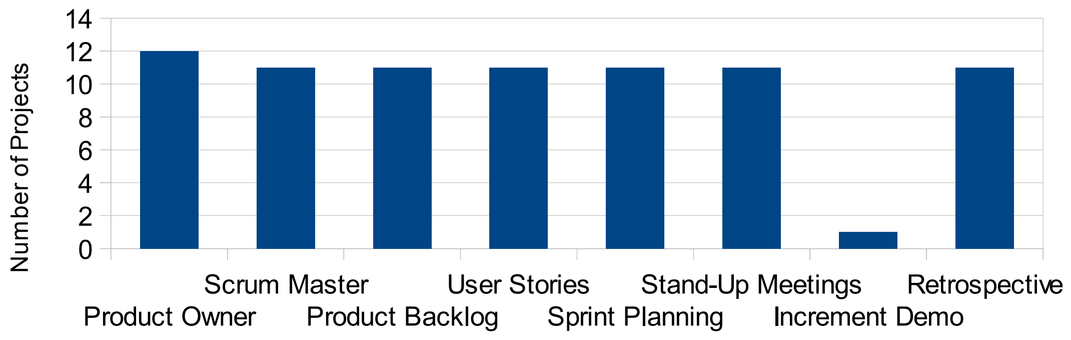

Figure 1. Findings Summary; Number of Projects Using Scrum Roles and Practices

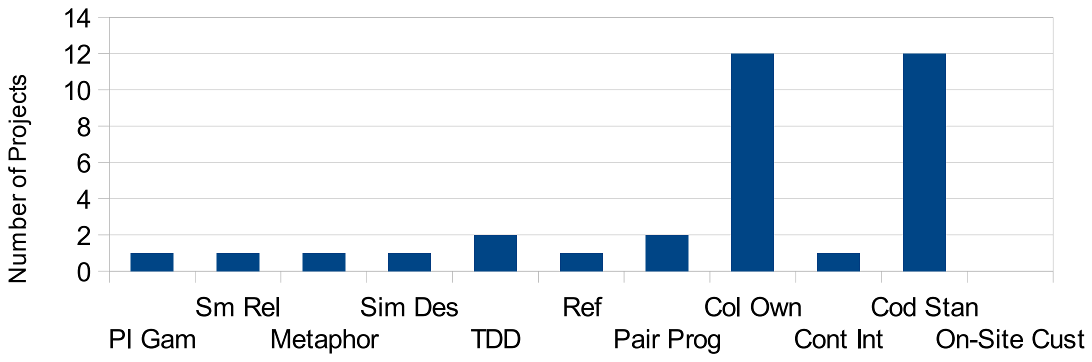

Figure 2. Findings Summary, Number of Projects Using XP Practices

programming techniques or different style of coding or anything like that

However, some form of two XP practices, collective code ownership and coding standards, are common practice on the enterprise software development investigated. The factors affecting adoption of some practices and not others is now discussed. The particular practices being selected and tailored and the ways the practices are being tailored is explored.

B. *Influences on Agile Practice Tailoring*

An important objective of this study is to understand the influences affecting tailoring of agile practices as much as their selection in enterprise projects. The main driver for this tailoring is embedding the software development practice into a wider enterprise context. Particularly those aspects of the enterprise context where project teams have little influence. The lack of influence tends to be over external organizational structures (such as external departments performing the build and release function) or quality assurance processes (such as procedures dictated for user acceptance testing of code, or deliverables that must be specific document formats for review by some high-level governance committee). Project teams can select agile practices for use within the team, that do not adversely impact or duplicate the work of external teams.

In addition, practices can be tailored for use in a distributed development environment. Arrangement of team members in certain time zones can dictate stand-up meetings that are conducted with video conference tools towards the end of their working day, rather than at the start.

I will now consider factors influencing selection of the anomalous agile practices. I will explore factors relating to the selection of scrum practices (1) sprints and sprint duration, (2) daily scrum and de-selection of the (3) increment demo.

In the subsequent paragraphs, I will then consider the de-selection of XP practices (4) test driven development, (5) pair programming, followed by discussion of the selection of (6) coding standards, (7) collective code ownership and (8) continuous integration.

1) *Sprints and Sprint Duration*

Sprints are used to focus the development team on production of usable code within a fixed time period. The sprints are un-interrupted and avoid demoralizing changes in product requirements.

Company B were using sprints lasting one month, in January 2010, with a product release cycle of six to nine months. By April 2011 they had reduced the product release cycle to 3-4 weeks. They have successfully moved to one month sprints producing and delivering production code that has been tested to full compliance with internal quality assurance processes. This reduction in release duration has resulted from a determined effort, supported by senior management, to reduce the time required to deploy production software. Further, substantial organizational changes have been required to enable such a software production cycle. Aligning sprint duration and release cycle reduction required the integration of requirements analysis as well as quality assurance processes with the shorter software development cycles. Thus, shorter development cycles must effectively cross traditional departmental boundaries.

The enhancements to the core CRM product of Company E, developed using the offshore team at Company D, use 14 day sprints. There are two teams working on the product, so each week there is either a sprint kick-off or a retrospective. The sprints are used to empower and motivate the development team. However, these increments are not deployed to the customer-base. The problem is linked to the scale and complexity of the product making continuous integration difficult to achieve, as will be discussed in (8) below.

2) *Daily Scrum*

The daily scrum is operated as a stand-up meeting on many of the projects investigated. The meeting were intended for team members to address three questions: (1) what have you done since the last scrum? (2) what will you do between now and the next scrum? and (3) what got in the way of doing work? This meeting style makes team members publicly (within the team, at any rate) accountable for their commitments.

A minority of the projects, had geographically dispersed team members, necessitating the use of video- or telephone-conferencing technology. The configuration of team members, in terms of their time zone, meant some team members joined the scrum towards the end, rather than at the beginning of their working day.

3) *Increment Demo*

To collect customer feedback, a demonstration of finished code at the end of a sprint is intended. However, in Company D code is collected into larger releases over a six to nine month timescale, since as has been seen it is too expensive to upgrade client's code base more frequently. In Company B on the other hand, there is little face-to-face contact with Internet customers. So, it is better to release new features into selected markets on an experimental basis.

Only Company F, in addition to the agile advocates at Company C, made effective use of Increment Demos.

4) *Test Driven Development*

It is argued that "any program feature that doesn't have an automated test, simply doesn't exist" [15]. Taken further, the suggestion is that it makes sense to write the tests first, and then fill in the code to pass the test. For example, in the Healthcare project in Company F, a conventional approach to Test Driven Development was used. The Scrum Master there states, *"when you have put the acceptance test case [initially] of course you know it is going to fail. Because we don't have a code yet, behind it."*

However, the other surveyed projects did not make much use of Test Driven Development. For example, according to an architect at Company D, "*what we did is more like real-time testing. It's not a test-driven approach. ... it was more like `I do this, okay, [then] I test this right now. I do that, [then] I test if that works'...*" Similarly, the Product Manager at Company B, stated "*[in relation to Test-Driven Development] we are aware of that, but we are not following that.*" Even the Agile advocates at Company C, had some difficulties, the project manager states, "*I wouldn't say we are rejected [Test Driven Development] but I would say we are on probably on different schemes, on different principles.*"

5) *Pair Programming*

In pair programming, all production code is written by two people looking at one computer screen. The person typing is thinking about the specific code context; the other person is thinking more strategically. Sometimes is it helpful to pair a novice with someone more experienced to provide support with a new task activity. The pairings are intended to be dynamic, swapping roles and pairs often.

It was surprising to find that only two of the projects used Pair Programming. For Company C the practice is a normal part of the their standard process, as envisaged in [15]. They encourage frequent rotation of pairs "*we try to constantly on rotate the pairs, on a daily basis. ... Every day, maximum we try to keep it as a couple of days beyond which we ask people to rotate.*"

In company D, however, Pair Programming is only used where the code is seen as unusually complex or difficult. According to a Scrum Master from Company D "*pair programming is one when we were initially on a spike, ... [where] you don't have much information, those are called spikes. So when we're doing that we do follow the - pair programming.*"

6) *Coding Standards*

All projects surveyed have coding standards. This results from the need to tailor practices to fit in with the enterprise development context. Enterprises often engage development team members from outside as well as within the organization, hence the need for an agreed coding standard to improve readability of code originating from different sources. Where multiple organizations are involved, common on large enterprise projects, projects must adopt shared policies. Sometimes shared policies belong to the client organization, sometimes one of the major international software service providers is brought on-board for their management and delivery expertise, so they will dictate project policies. Or, they may be dictated by the client or the lead software development organization or by the quality assurance standards adopted for the project.

The standards adopted may be the result of an ad hoc selection process for a given project. Where technical thought leaders have a choice they prefer light touch coding standards, which are sometimes public domain, to encourage widespread compliance within the team. The standards are explicitly to support knowledge transfer, induction of new joiners and collaborative working in general within the team.

7) Collective Code Ownership

Enterprise software development teams typically operate with some form of collective code ownership. Software considered "complete" by the development team must be handed over to external user acceptance test teams for detailed examination and regression testing. Subsequently, after user acceptance tests are passed, software is handed over again. This time the software is given to external release teams for deployment and maintenance. Thus, collective code ownership is forced upon project teams and developers by the presence of external build and release, quality assurance and user acceptance test teams. According to a team member at Company D "*we finish the development and we send it to a team called the Sustaining Team and they do the proper testing and maintenance*" and further "*they talk directly to the clients. If the clients, after taking release, have a few problems, it's them who will fix actually everything.*"

8) Continuous Integration

Advocates of continuous integration argue for daily builds of working, shippable, software [15]. New features are added to the integration progressively during the working day. As soon as the new features are added, they are tested to resolve any clashes with code elsewhere.

Continuous integration is challenging in enterprise software development projects because of the size, scale and complexity of the system under development. According to the Director of Engineering at Company E, "*regression tests take too long to do it every sprint. So we don't run a regression suite, it is like three and a half days to run a full regression suite so we don't run that* [every Sprint]*.*" The release process is described in more detail by the architect at Company D:

> *we have three cycles [1] from the code freeze date to integration complete, there will be no check-ins and check-outs from the [code] repository. And then [2] from integration complete to release [candidate]. So again the same test cases would be repeated... And then the final phase [3] is when... the CD is built properly and it can be installed from the CD and then release it.*

Clearly, this is a much more elaborate process than the daily builds and release of code to clients at the end of each sprint envisaged in [12]. So, according to the Director of Engineering at Company E "*we don't necessarily have a runnable application at every possible opportunity.*" He elaborates:

> *...people begin to see there is no take up of releases, if they are too frequent. Because the effort to upgrade from A to B is so costly that it is not practical for people taking new releases on a regular basis. So spitting out new releases, some of them may never get taken up unless you sell something* [to a new client].

There is extensive customization of software to suit individual client enterprises, in this business domain. Migration to a new release could entail significant and expensive rework of customizations in order to upgrade customer code. Thus, in this domain, outputs from successive sprints are collected to form half-yearly releases.

IV. Discussion

This research found that most of the projects investigated used scrum agile practices, confirming the findings from the project reported in [7]. These enterprises use sprints, backlogs and scrum roles such as product owner, scrum master and team members. Team members were self organizing and empowered to estimate and assign work durations to tasks and to select tasks for their own work.

Only one of the companies routinely used XP practices. The other projects were not using most XP style agile practices, such as pair programming, test driven development, simple design, metaphors, re-factoring and so on. These techniques are either not seen as important or are difficult to implement in an enterprise context. Manual methods such as story boards for managing user stories, while still favored for their simple ability to communicate effectively to co-located team members, are being superseded by automated software tools.

The sprint is readily adopted in enterprise software development settings, since it can be internal to the development team. The program manager in Company E described a CRM project development team, in the USA, using short increments while embedded within a client enterprise that organized departments around a waterfall development tradition.

Some of the sprint practices were adapted for the enterprise context. So Company E, and their offshore software service provider Company D, used 14 day sprints even though usable code was not shipped to customers at the end of each increment. The productivity and morale benefits were seen to be attractive in themselves.

The scrum is easily adopted, in an enterprise setting, since it can be conducted internally, possibly even surreptitiously, within the development team. In Company B the scrum evolved from being internal to the team, in January 2010, to involving external requirements analysis, quality assurance and testing stakeholders in April 2011.

In several of the enterprise projects investigated the increment demo did not play an important role, either because there was no intention to release code at the end of each sprint or because face-to-face contact with customers is difficult.

Organizational boundaries made test driven development unattractive, since corporate quality assurance processes relied on user acceptance testing being performed by independent and sometimes external teams. A major organizational change process was required in Company B to align sprint durations and production release cycles. In Company D source code was handed to quality assurance teams for time consuming regression testing.

It is puzzling that greater use was not made by the projects of pair programming. There do not appear to be organizational boundaries hindering adoption. The use of pair programming in enterprise software teams was limited to unusual, exploratory or complex code. Only the agile advocates at Company C made use of pair programming as standard practice.

In contrast, coding standards are rather enforced upon enterprise development teams by the large organization setting. The plethora of project stakeholders and employment arrangements for project team members make standards desirable to ensure readability of code. Similarly, there is a general recognition that source code needs to be shared with other project stakeholders.

The size and complexity of enterprise software products make continuous integration a challenge. Daily cycles of build, integrate and test seem attractive. But, the three days required by Company E to run regression tests is typical of the challenge. Elaborate cycles of code freezing, regression testing and bug fixing is performed before adding installation code and cutting to CDs for distribution. These tasks are performed by a separate specialist build and release department.

So, these findings are at variance with some earlier sources. The projects I investigated did not make extensive use of continuous integration, pair programming or test driven development in contrast with the findings of [20]. Further, I found support for less studied practices such as user stories, burn-down charts and planning meetings. I also found some (possibly distorted implementations of) agile practices that were well-established in large enterprises. For example, all the large enterprises had their own coding standards, or adopted publicly available standards to facilitate sharing of code artifacts between distributed team members. Further, common ownership of code is forced upon the teams by the logistics of independent and often external release management and user acceptance testing teams.

Early writing on XP was explicit that to achieve the full benefits of adoption practices should be implemented wholesale (for example [15] page 149). Further, there was emphasis on the physical arrangement of co-located work areas ([15] page 77) and customer representative working physically along side the development team in the form of an "on-site customer" ([15] page 60). Clearly these aspects of agile development are not well-placed for adoption by distributed geographically teams.

The adoption of agile practices in the surveyed enterprises was hindered by complex organizational as well as geographical boundaries. The existence of corporate project management standards, hierarchically formulated build and release teams that enforce enterprise quality standards apparently prohibit the use of practices such as test driven development. Some projects have adopted such practices, surreptitiously, out of belief in the productivity benefits. These projects have attempted to gain benefits from agile methods despite corporate constraints.

The research also shows that agile adoption can create a ripple effect of change through the enterprise. Company B in the Internet sector has succeeded in reducing release cycles from six or nine months to three or four weeks, but this involved drawing external quality assurance teams into the Sprint teams.

V. Limitations

This type of study relies on self-reported measures. This approach risks being deflected by respondents that have their own motives for over- or under-reporting issues. Data triangulation, obtaining different stakeholders perspectives, and method triangulation, such using as using documentary as well as interview sources, can improve reliability of findings. There is a good level of data triangulation in the investigation of Company E and their offshore software service provider Company D. Sources range from the Director of Engineering to scrum masters and team members. Research rigor could be improved by a greater effort to triangulate data collection from a range of stakeholders within other projects. Although, the difficulty of getting access to practitioner teams has been mentioned.

The results should not be generalized to different industry sectors. For example, embedded software system developers in the construction industry work under profoundly different commercial lead-times and are accountable to external planning authorities. I expect the influences on agile practice adoption in the construction industry would be very different, yet surveys such as [9] seem to unhelpfully conflate results from widely differing industry sectors.

VI. Conclusions and Further Work

This research explores enterprise software development in a climate of downward pressure on prices from clients, increasing competition and reluctance to invest unless projects demonstrate customer value. Enterprise software development is conducted on a global scale, with offshore or near-shore partnerships becoming the norm.

I had expected to find a steady process of agile practice adoption from proof of concept projects and smaller collocated XP software development teams. Instead, the adoption of scrum roles and practices was observed. Short iterations, even if they did not deliver production code to customers, were used to improve team morale and focus attention on customer needs. Short daily standup meetings conducted using telephone and video conference technologies, if necessary, were considered helpful.

In contrast, software development practices from the XP method were much less widely adopted. Test driven development did not integrate well with established enterprise quality assurance policies. Separate build and release teams, responsible for user acceptance testing, did not integrate into the short release cycles in some of the companies investigated.

Pair programming was only used in special circumstances, such as where code was new and complex. There was no obvious problem with integrating this practice into the enterprise. Pair programming could be performed without negatively impacting the work of any other departments. Yet it was puzzling to find that there was not more widespread adoption.

However, coding standards and collective code ownership are normal practice on large scale projects. There was recognition that common standards are necessary in large and diverse project teams and that code will need to be shared with others during the project lifetime. Continuous integration was found to be challenging because of the time required to run regression tests in large and complex enterprise software products.

This study can be further developed in a number of ways. Further data gathering, over time, would provide evidence of longitudinal change in adoption patterns. In a different direction, it is planned to use theory from the field of organizational studies to explore the socio-technical factors affecting agile practice adoption and tailoring in more detail.

It is puzzling that XP practices, notably pair programming, that do not impinge on any broader enterprise context are not more widely used. Further exploration is required to understand the lack of awareness or lack of commitment by development teams to these practices.

Acknowledgments

I am grateful to all the companies and interviewees who were generous enough to contribute their time and resources to participate in this research. Thanks also to the current and former students of the Executive MBA at the Indian Institute of Management, Bangalore who helped identify target companies. The research benefited in part from travel funding from the UK Deputy High Commission Science and Innovation Network.

References

[1] W. Aspray, F. Mayadas, and M. Y. Vardi, Eds., “Globalization and Offshoring of Software: A Report of the ACM Job Migration Task Force.” ACM, 2006.

[2] J. D. Herbsleb and D. Moitra, “Global software development,” *Software, IEEE*, vol. 18, no. 2, pp. 16 -20, Apr. 2001.

[3] P. Joannou, “Enterprise, Systems, and Software Engineering–The Need for Integration,” *Computer, IEEE*, vol. 40, no. 5, pp. 103 -105, May 2007.

[4] A. Bartels and J. R. Rymer, “The Future Of Enterprise Software: Market Overview.” Forrester Research, Inc, Jun-2006.

[5] M. Lindvall et al., “Agile software development in large organizations,” *Computer, IEEE*, vol. 37, no. 12, pp. 26 - 34, Dec. 2004.

[6] B. Boehm and R. Turner, “Management challenges to implementing agile processes in traditional development organizations,” *Software, IEEE*, vol. 22, no. 5, pp. 30-39, 2005.

[7] L. Pries-Heje and J. Pries-Heje, “Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India,” in *AGILE Conference (AGILE), 2011*, 2011, pp. 20 -28.

[8] “Agile Alliance.” [Online]. Available: http://www.agilealliance.org/. [Accessed: 25-Sep-2011].

[9] S. de Cesare, M. Lycett, R. D. Macredie, C. Patel, and R. Paul, “Examining Perceptions of Agility in Software Development Practice,” *Communications of the ACM*, vol. 53, no. 6, pp. 126-30, Jun. 2010.

[10] T. Dyba and T. Dingsoyr, “What Do We Know about Agile Software Development?,” *Software, IEEE*, vol. 26, no. 5, pp. 6 -9, Oct. 2009.

[11] K. Schwaber, *Agile Project Management with Scrum*. Redmond, WA.: Microsoft Press, 2004.

[12] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*. Addison Wesley, 2009.

[13] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[14] C. Larman, *Agile and Iterative Development: A Manager’s Guide*. Pearson Education, 2003.

[15] K. Beck and C. Andres, *Extreme Programming Explained*, 2nd ed. Addison Wesley, 2004.

[16] J. D. Herbsleb, “Global Software Engineering: The Future of Socio-technical Coordination,” in *Future of Software Engineering, 2007. FOSE ’07*, 2007, pp. 188 -198.

[17] J. A. Espinosa and E. Carmel, “The impact of time separation on coordination in global software teams: a conceptual foundation.,” *Software Process: Improvement & Practice*, vol. 8, no. 4, pp. 249 -266, 2003.

[18] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood, “A review of awareness in distributed collaborative software engineering,” *Software: Practice and Experience*, vol. 40, no. 12, pp. 1107–1133, 2010.

[19] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in a globally distributed project: a case study,” *Software Process: Improvement and Practice*, vol. 13, no. 6, pp. 527–544, 2008.

[20] S. Jalali and C. Wohlin, “Agile Practices in Global Software Engineering - A Systematic Map,” in *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, 2010, pp. 45 -54.

[21] E. Hossain, M. A. Babar, and H.-young Paik, “Using Scrum in Global Software Development: A Systematic Literature Review,” in *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, 2009, pp. 175 -184.

[22] R. K. Yin, *Case Study Research: Design and Methods*, 4th ed. Thousand Oaks, California: Sage Publications, Inc, 2008.

[23] M. Q. Patton, *Qualitative Research & Evaluation Methods*, 3rd ed. Thousand Oaks, California: SAGE Publications, Inc, 2001.

Appendix I Interview Guide

A. Motivation and Purpose of Research

I want to ask you about your experience of geographically distributed agile software development projects. The research involves interviews with people doing a range of different roles and from companies with different development models.

I want to learn more about your views of agile processes. I am particularly interested to know what factors are affected by geographical location and separation. The purpose here is to try to understand the factors that affect project outcomes, successful or otherwise, so that we can try to learn for the future.

B. Ethical Commitments and Informed Consent

I want to ask you the following questions and tape record your answers.

I will keep your responses absolutely confidential. Certainly nothing will be shared with any client companies. I do plan

to publish interview extracts but I will make names and companies anonymous.
Can I switch on the recorder?

C. Your Current Project(s)
How many projects are you working on currently?
What is (was) the title of your current (or most recent) project?
What is the project management structure?
How is the project organized geographically?
How many people are in the project team?

D. Agile Practices
What Agile practices do you advocate for offshore projects?
What agile practices do you avoid or not recommend?

E. Requirements
How are requirements decided and prioritized?
How do user stories evolve over time?
How do user stories move up or down the backlog?

F. Product Owner/Customer (POC)
How do you represent the product development team within the client organization?
How do you represent the client organization within the product development team?

G. Releases and Testing
How do unit tested code become a release?
How is user acceptance testing managed?
How are bugs reported back, prioritized and fixed?

H. Challenges
What challenges do you face in agile offshore projects?
How have you tried to address these challenges?
What challenges remain to be resolved?

I. Learning
How does learning take place within the team?
How does learning take place for you personally?

J. Any other comments
Do you have any further comments in relation to geographically distributed agile development projects?